University at Buffalo
Department of Computer Science and Engineering
School of Engineering and Applied Sciences

Project-3: Intro to Machine Learning

Team members: Abhiram Yadu Sribhashyam (50560760), Sri Gayatri Morapakala (50537847)

Brief Overview of the Dataset:

The **MNIST** dataset, or Modified National Institute of Standards and Technology database, is a widely used collection of handwritten. It is essentially the "Hello World" for machine learning image processing and serves as a benchmark for many algorithms. A few characteristics of the dataset are:

Data: It consists of 70,000 grayscale images of handwritten digits (0-9), each 28x28 pixels.

Structure: There are 60,000 images in the training set and 10,000 in the test set. Each image is labeled with the corresponding digit it represents.

The original data comes from NIST (National Institute of Standards and Technology) but was specifically selected and preprocessed for machine learning research. This "remixing" process involved selecting a subset of digits and standardizing factors like size and centering.

The MNIST dataset's popularity can be attributed to several key factors:

- MNIST offers a well-defined task (handwritten digit recognition) with a manageable amount of preprocessed data. This makes it ideal for beginners to experiment with image processing and machine learning concepts without getting bogged down in complex data collection or pre-processing steps.
- MNIST serves as a common ground for researchers and developers to compare the performance of various machine learning algorithms on a consistent dataset. This allows for fair evaluation and easier identification of the best-performing models for similar tasks.
- Due to its simplicity, MNIST provides a solid foundation for building more complex models.
- MNIST is readily available online and free to use, removing barriers to entry for anyone interested in exploring machine learning.

For this project, we have chosen three distinct machine-learning models to tackle the problem of digit classification: Convolutional Neural Networks (CNN), k-Nearest Neighbors (k-NN), and Support Vector Machines (SVM).

Each of the above models offers unique strengths: CNNs are renowned for capturing spatial hierarchies in image data; k-NN is a simple, instance-based learning algorithm effective for small datasets; and SVMs are powerful for achieving high accuracy with an appropriate kernel. CNNs are known to perform better on MNIST as:

- CNNs excel at capturing spatial relationships between features in images. MNIST digits, despite variations in handwriting, share a basic spatial structure (e.g., a straight line for 1, a closed loop for 0). CNNs can effectively learn these spatial patterns to differentiate between digits.

- Through its convolutional layers, a CNN can automatically learn the relevant features from the images (like edges, and curves) that are crucial for digit recognition. MNIST provides a good starting point for this learning process.

Our selection is motivated by the desire to compare a deep learning approach (CNN) with more traditional machine learning techniques (k-NN and SVM), providing a comprehensive view of the problem's landscape. Additionally, we aim to explore the impact of different hyperparameters and activation functions on the models' performance. By examining the nuances of these models' capabilities and limitations through the lens of the MNIST dataset, we aim to gain deeper insights into the nature of digit classification tasks and the factors contributing to model success.

Data Processing:

The MNIST dataset was loaded using Keras, and subsets for each class were created to ensure a balanced representation in the training and validation sets. 50,000 images were used for training and 10,000 for validation, with each class represented equally. The input data underwent several preprocessing steps to facilitate model training:

- Reshaping: The images were reshaped to 28x28x1 to comply with CNN input requirements and flattened to 784 features for k-NN and SVM.
- Normalization: Pixel values were normalized to the [0,1] range to improve the numerical stability of our models and hasten convergence.
- One-Hot Encoding: The target labels were one-hot encoded to suit the categorical cross-entropy loss function used in the CNN.

Algorithms Used:

CNN: The two CNN models consisted of sequential layers designed to capture hierarchical patterns. The first convolutional layer with 32 filters was followed by a max-pooling layer, intended to reduce the spatial dimensions while retaining the most salient features. The second convolutional layer with 64 filters increased the depth of the network, allowing for the capture of more complex features. Two dense layers followed with 64 neurons each, introducing the capacity to learn non-linear combinations of the high-level features extracted by the convolutional layers. The output layer employed a softmax activation function to yield a probability distribution over the 10 classes.

- Parameters: Sigmoid and ReLU activations were compared. The ReLU-activated model performed better, likely due to mitigating the vanishing gradient problem. The network used an SGD optimizer, a common choice for training neural networks.

k-NN: This non-parametric model was a straightforward implementation utilizing the standard Euclidean distance to measure the similarity between instances. It did not require a learning phase, which made it much faster to initiate compared to the CNN. Its prediction phase was computationally heavy, involving distance calculations between the test instance and all training instances.

- Parameters: The number of neighbors was set to 3 after preliminary tests showed that this provided a good balance between overfitting and underfitting.

SVM: SVM models were trained using both linear and RBF kernels. The RBF kernel SVM is particularly suited for non-linear problems, as it can map input features into high-dimensional space where the data points become more separable.

- Parameters: The 'C' parameter, which dictates the trade-off between the decision boundary margin and misclassification rate, was left at default values. Exploring its tuning might yield further insights into the model's performance.

Empirical Results and Statistical Analysis:

For the CNN models, the sigmoid-based network showed slower convergence and lower final accuracy compared to the ReLU-based network. This aligns with expectations as ReLU activation tends to alleviate the vanishing gradient problem.

The k-NN model achieved respectable accuracy, with the simplicity of the model offering quick iteration times. However, the prediction phase was slower compared to CNN, as expected due to the lazy learning approach of k-NN.

The SVM models exhibited a stark performance contrast. The linear kernel SVM trained faster but had lower accuracy than the RBF kernel SVM, which was computationally intensive but achieved a higher accuracy, indicating a more complex decision boundary.

The ROC and AUC curves revealed that the CNN and SVM with the RBF kernel managed to achieve impressive class separation, while the k-NN, despite its simplicity, showed a decent performance across all classes.

Model Comparison:

After careful evaluation, the CNN model with ReLU activation was chosen as the superior model due to several key factors:

- **Performance**: It achieved the highest accuracy on the validation set and demonstrated clear learning progress as visualized in the loss and accuracy graphs.
- **Generalization**: The model showed signs of better generalization to unseen data, as indicated by the convergence of training and validation accuracy.
- **Computational Efficiency**: Although more computationally intensive during the training phase than k-NN, the CNN with ReLU activation proved to be less resource-intensive than the RBF-kernel SVM while providing superior accuracy.

The CNN model with ReLU activation is the model of choice. It offered an optimal blend of high accuracy and generalization capabilities while still being computationally feasible. The chosen parameters, such as the use of ReLU and a standard SGD optimizer, were validated by their widespread success in various image recognition tasks in the literature and by the empirical results obtained during the experiments.

Justification of our Approach:

Our approach to the classification task was driven by the desire to contrast the capabilities of a deep learning architecture with traditional machine learning models. CNNs, with their depth and complexity, are well-suited to capture the hierarchical patterns in image data. The ReLU activation function's non-saturating nature helped prevent vanishing gradients, leading to better performance than the sigmoid-based CNN.

In k-NN, the choice of k=3 was a balance between overfitting and underfitting and was computationally feasible. The Euclidean distance is a natural choice for image data represented in vector form.

For SVMs, the use of RBF and linear kernels aimed to explore the trade-offs between computational complexity and decision boundary refinement. The RBF kernel's higher dimensionality mapping proved beneficial for capturing more complex patterns in the data.

Conclusion:

The results of our experiments highlight the trade-offs between computational complexity and classification accuracy. While CNNs and SVMs with RBF kernels demonstrated high accuracy, they did so with increased training times. The k-NN model, while less accurate, provided quick training and could serve as a baseline for more complex models.

For future work, we would explore the impact of further hyperparameter tuning, the incorporation of regularization techniques to prevent overfitting, and the use of ensemble methods to potentially improve upon the results obtained by individual models.

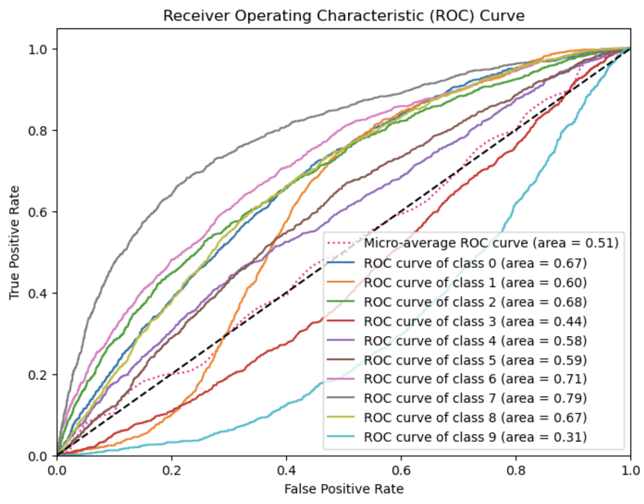
Visualizations:

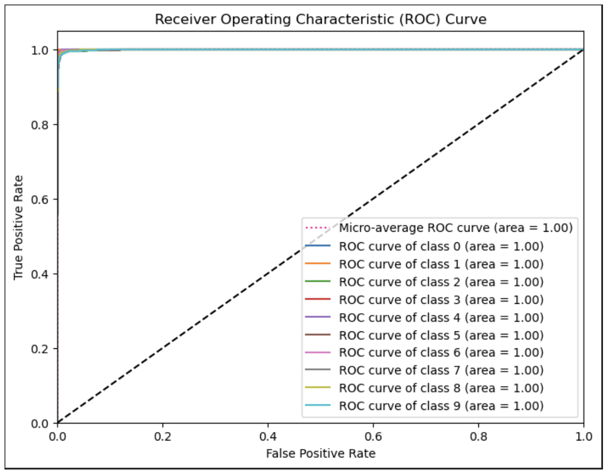
Visualization of each model's performance is illustrated by the following diagrams and plots.

- CNN: Plots of training and validation loss and accuracy across epochs for models with sigmoid and ReLU activations.
- k-NN: A confusion matrix displaying the prediction results across different classes, ROC curves for each class with their respective AUC scores, and Precision-Recall curves to evaluate the model's precision at various threshold settings.
- SVM: Confusion matrices for SVM models with different kernels, and ROC curves for the SVM with the RBF kernel to illustrate its class separation capability.

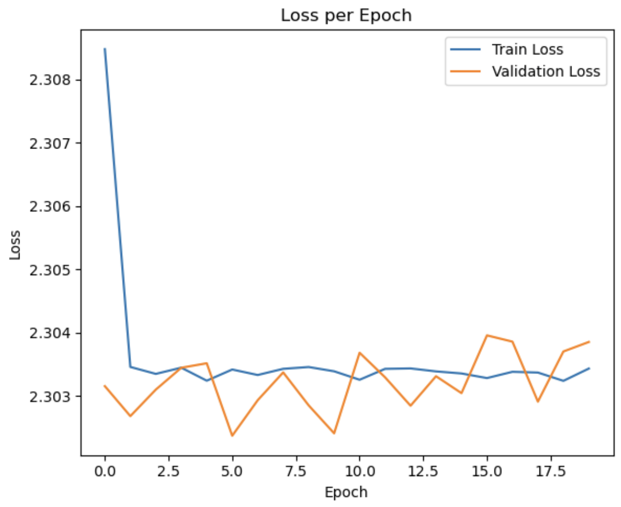
ROC for Sigmoid

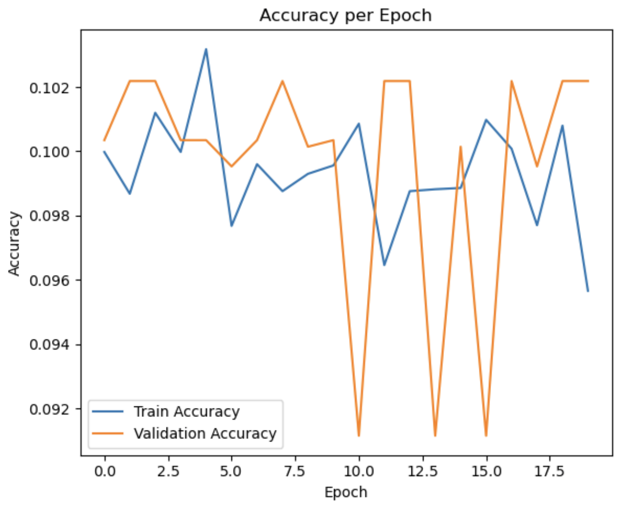
ROC for ReLU



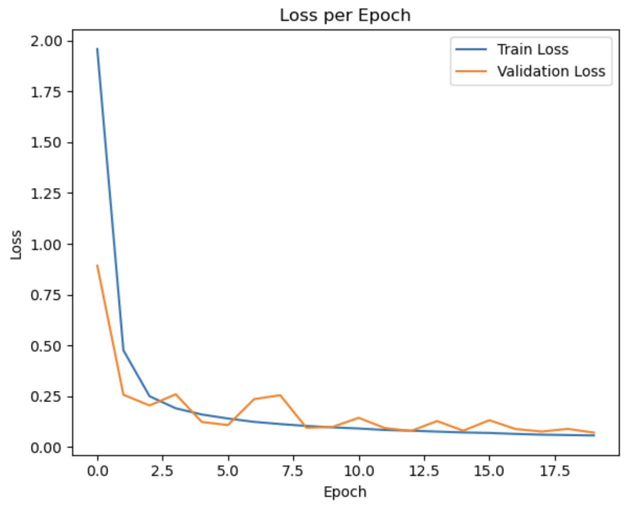


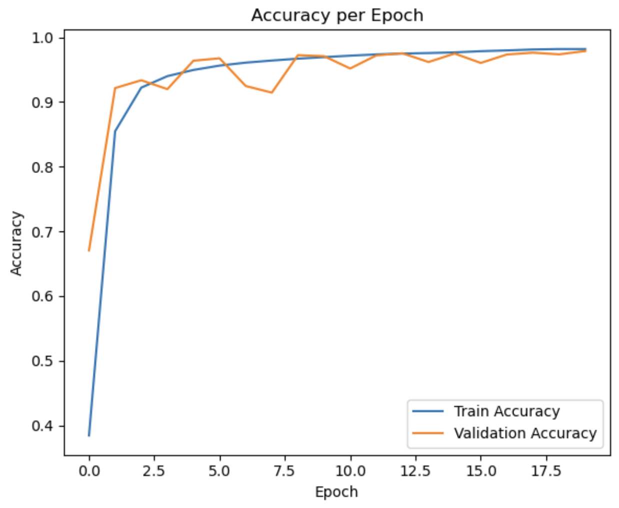
Below Plot shows Loss and Accuracy per Epoch for the Sigmoid Activation Function

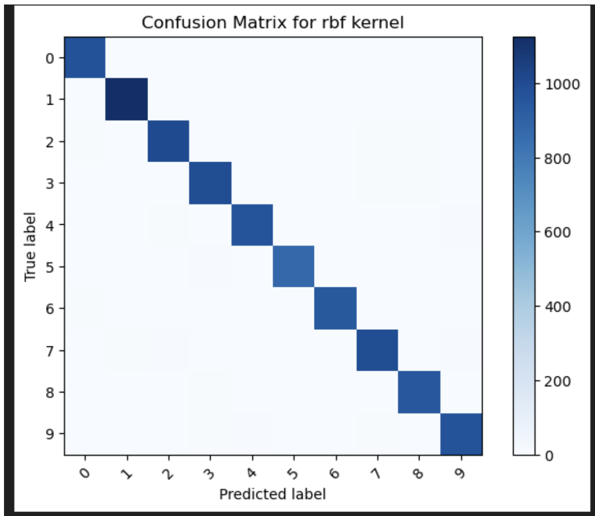




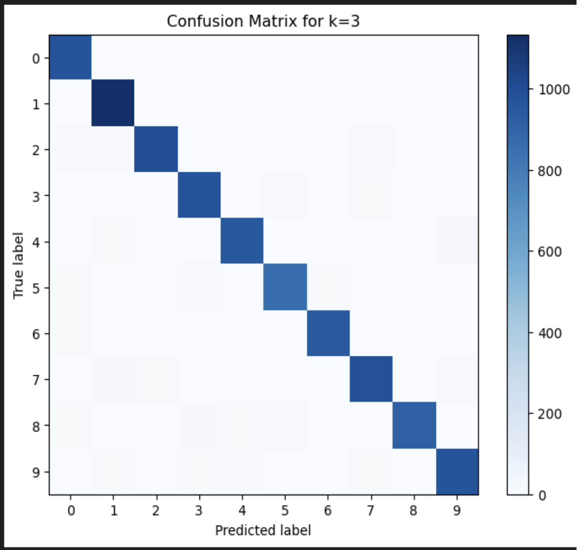
Below Plot shows Loss and Accuracy per Epoch for the ReLU Activation Function

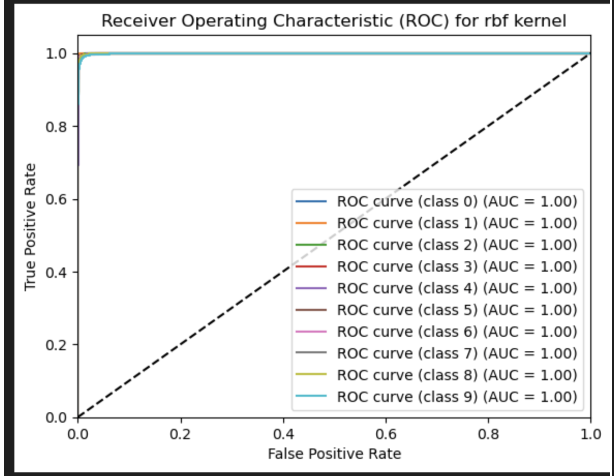


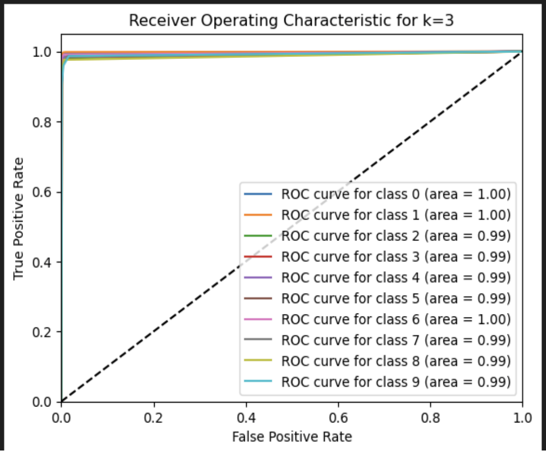




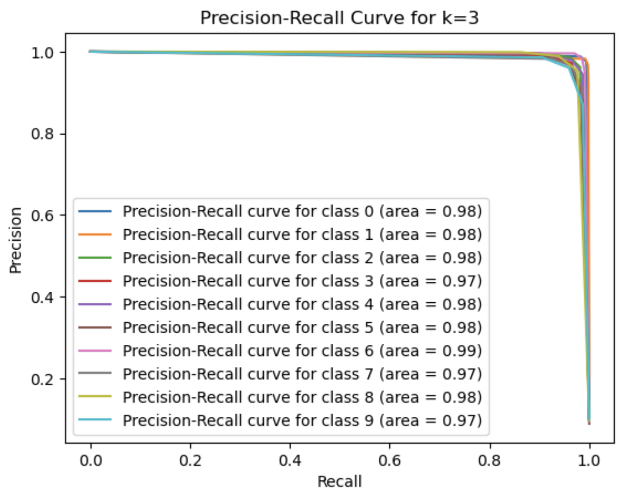
```
Accuracy for k=3: 0.97
```

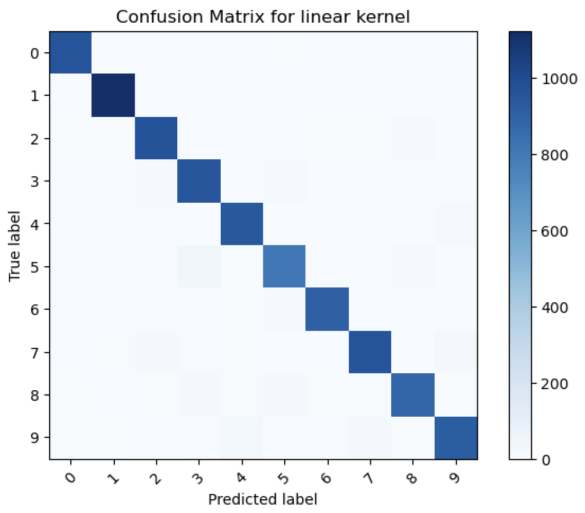






```
Accuracy for linear kernel: 0.94
```


Precision-Recall Curve for k=3
Precision
Recall
Precision-Recall curve for class 0 (area = 0.98)
Precision-Recall curve for class 1 (area = 0.98)
Precision-Recall curve for class 2 (area = 0.98)
Precision-Recall curve for class 3 (area = 0.97)
Precision-Recall curve for class 4 (area = 0.98)
Precision-Recall curve for class 5 (area = 0.98)
Precision-Recall curve for class 6 (area = 0.99)
Precision-Recall curve for class 7 (area = 0.97)
Precision-Recall curve for class 8 (area = 0.98)
Precision-Recall curve for class 9 (area = 0.97)


Confusion Matrix for linear kernel
True label
Predicted label